11-22-06

AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE	
APPEAL BRIEF	Atty. Docket No. **VIGN1690-3**

Applicant **Dean Moses et al.**	
Application Number **10/091,486**	Date Filed **03/07/2002**
Title **Method and System for Deploying Web Components Between Portals in a Portal Framework**	
Group Art Unit **2154** -	Examiner **Donaghue, Larry D.**
Confirmation Number: **9466**	

Mail Stop: Appeal Brief- Patents

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Dear Sir:

Certification Under 37 C.F.R. §1.10

I hereby certify that this document is being deposited with the United States Postal Service as Express Mail No. **EV887561842US** in an envelope addressed to: Appeal Brief, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313 on 11-21, 2006.

Julie H. Blackard
Signature

JULIE H. BLACKARD
Printed Name

Further to the Notice of Appeal filed September 21, 2006, the Appellant presents this Appeal Brief. The Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

11/24/2006 DEMMANU1 00000019 10091486
01 FC:1402 500.00 OP

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The subject application is owned by Vignette, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at Corporation Trust Center, 1209 Orange Street. Wilmington, DE 19801.

II. RELATED APPEALS AND INTERFERENCES

The Appellant believes that there are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-46 were originally filed in the present application with Claims 1, 8, 11, 16, 23, 26, 30, 39, and 43 being independent. Claim 33 was cancelled and Claims 47-58 were added by the Appellant's Response to Office Action dated August 13, 2003. Claims 2, 8-10, 23-25, 39-42, 44-45, and 47-58 were cancelled and Claims 59-78 were added by the Appellant's Response to Office Action dated December 12, 2003. Claims 79-88 were added by the Appellant's Response to Office Action dated May 19, 2004. The Appendix hereto reflects the current status of Claims 1-88: Claims 2, 8-10, 23-25, 33, 39-42, 44-45, and 47-58 are cancelled and Claims 1, 3-7, 11-22, 26-32, 34-38, 43, 46, and 59-88 stand rejected under 35 U.S.C. §103(a). Claims 1, 3-7, 11-22, 26-32, 34-38, 43, 46, and 59-88 are being appealed.

IV. STATUS OF AMENDMENTS

No amendments were filed subsequent to the Final Office Action dated June 28, 2006. The first amendment to the application, filed November 11, 2003, amended Claims 1-2, 8, 10-12, 16, 23, 25-27, 30-31, 39, 41, 43, and 45 and cancelled Claim 33. The second amendment to the application, filed March 5, 2004, amended Claims 1, 3-4, 6-7, cancelled Claims 2, 8-10, 23-25, 39-42, 44-45, and 47-58, and added Claims 59-78. The third amendment to the application, filed August 19, 2004, amended Claims 3-4, 46, and 71 and added Claims 79-88. The fourth amendment to the application, filed January 18, 2006, amended Claims 1, 3, 11, 16, 26, 30-31, 43, 60, 72, 79, 83, and 87.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent Claims 1, 11, 16, 26, 30, 43, 79, 83, and 87 are involved in the present appeal. The Appellant respectfully submits the following concise explanation of the subject matter defined in each of the independent Claims 1, 11, 16, 26, 30, 43, 79, 83, and 87.

Claim 1 recites:

> A computer-implemented method of deploying components of a site between systems, the method comprising the steps of:
>
> storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;
>
> transferring the individual export file to a system at a remote location, wherein the system is a web portal capable of executing program instructions; and
>
> extracting the assets from the individual export file to a plurality of locations on the system at the remote location, wherein the assets include file assets and non-file assets configured to operate on the web portal at the remote location.

The method of Claim 1 deploys components of a site between systems in a portal framework. *See* Application, page 3, lines 8-10. As exemplified in FIGURE 1, a portal framework ("Framework") 100 may comprise systems 102, systems 110, systems 106, and systems 108. In a preferred embodiment, systems 102 implement user systems, systems 110 implement web servers, systems 106 implement application servers, and systems 108 implement database systems. A web server 110 and an application server 106 can be one in the same computer system. *See* Application, page 13, lines 8-9.

The method of Claim 1 includes "storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects." A site is considered, for the purposes of the present invention, to be a collection of software objects given a single identity. *See* Application, page 9, lines 14-15. The single identity may be characterized by a shared look-and-feel, a shared set of navigation links, and members of a group who are automatically granted privileges to perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 9, lines 15-19. The look-and-feel of a site is referred to, for purposes of the present invention, as the branding for the site. *See* Application, page 51, lines 17-19. Components (objects) of a site include file assets and non-file assets encapsulating elements of the site (e.g., logon

permissions, administrative permissions, site branding, site content and site navigation, etc.). *See* Application, page 9, line 14, through page 10, line 21. Components of a site may be designated for export as an individual export file. *See* Application, page 3, lines 12-13. According to the method of Claim 1, the assets of at least one component (object) of a site that is designated for export is packaged in an individual export file transferable between systems (portals) in the portal framework.

The method of Claim 1 includes "transferring the individual export file to a system at a remote location, wherein the system is a web portal capable of executing program instructions." The collection of software objects may be managed by a set of users granted privileges associated with respective objects in the collection of software objects. *See* Application, page 8, lines 10-14. Following the example shown in FIGURE 1, Framework 100 may employ and maintain portals to provide gateways for access to the collection of software objects of a site based on granted privileges. *See* Application, page 8, lines 14-15. Through these portals, users with proper privileges can create and manage the collection of software objects of a site and perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 8, lines 10-14, FIGURES 16a-16b. Administration of sites may be performed anywhere in Framework 100. *See* Application, page 34, lines 4-16. For example, a user system 102 may be used to provide for the administration of sites on Framework 100. *See* Application, page 12, lines 4-9. As another example, the presentation of web site and administration of sites objects may be implemented by a system 106. *See* Application, page 13, lines 1-6. The branding for the site may apply to all users in Framework 100. *See* Application, page 62, lines 11-13. Examples of site branding are shown in FIGURES 18 (an end user site, *see also* Application, page 63, line 18, through page 66, line 3) and 20a-20b (an administration site, *see also* Application, page 56, lines 1-8). Thus, according to the method of Claim 1, the assets of at least one component (object) of a site that is designated for export is packaged in an individual export file and transferred to a system at a remote location within the portal framework.

The method of Claim 1 includes "extracting the assets from the individual export file to a plurality of locations on the system at the remote location, wherein the assets include file assets and non-file assets configured to operate on the web portal at the remote location." The exported assets include file assets and non-file assets configured to operate on the system at the remote location. *See* Application, page 4, lines 14-15. File assets include resources such as

code including JSP pages, ASP pages, Java classes and/or object oriented programming language classes, and images including GIF files, etc. *See* Application, page 10, lines 5-7. Non-file assets include instantiated (non-static) programming language objects, permissions, user preferences and setting such as users, groups, modules, module types, pages, menus, themes, structures, styles and templates. *See* Application, page 9, line 19, through page 10, line 2, and lines 8-11. Thus, according to the method of Claim 1, the assets of at least one component (object) of a site that is designated for export is packaged in an individual export file, transferred to a system at a remote location within the portal framework, and extracted from the export file to a plurality of locations on the system at the remote location. *See* Application, page 3, lines 17-18.

Claim 11 recites:

> A computer-implemented method of importing components of a site to a system at a remote location in a portal framework, the method comprising the steps of:
> extracting at least one object of a component from an individual export file of an exported site, wherein the exported site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;
> wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location; and
> storing each object of the component to a location on the system at the remote location.

The method of Claim 11 imports components of a site to a system at a remote location in a portal framework. *See* Application, page 3, lines 8-13. As exemplified in FIGURE 1, a portal framework ("Framework") 100 may comprise systems 102, systems 110, systems 106, and systems 108. In a preferred embodiment, systems 102 implement user systems, systems 110 implement web servers, systems 106 implement application servers, and systems 108 implement database systems. A web server 110 and an application server 106 can be one in the same computer system. *See* Application, page 13, lines 8-9. A site is considered, for the purposes of the present invention, to be a collection of software objects given a single identity. *See* Application, page 9, lines 14-15. The single identity may be characterized by a shared look-and-feel, a shared set of navigation links, and members of a group who are automatically granted privileges to perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 9, lines 15-19. The look-and-feel of a site is referred to, for purposes of the present invention, as the branding for the site. *See* Application, page 51, lines 17-19. The collection of software objects may be managed by a set of users granted privileges associated with respective objects in the collection of software

objects. *See* Application, page 8, lines 10-14. Following the example shown in FIGURE 1, Framework 100 may employ and maintain portals to provide gateways for access to the collection of software objects of a site based on granted privileges. *See* Application, page 8, lines 14-15. Through these portals, users with proper privileges can create and manage the collection of software objects of a site and perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 8, lines 10-14, FIGURES 16a-16b. Administration of sites may be performed anywhere in Framework 100. *See* Application, page 34, lines 4-16. For example, a user system 102 may be used to provide for the administration of sites on Framework 100. *See* Application, page 12, lines 4-9. As another example, the presentation of web site and administration of sites objects may be implemented by a system 106. *See* Application, page 13, lines 1-6. The branding for the site may apply to all users in Framework 100. *See* Application, page 62, lines 11-13. Examples of site branding are shown in FIGURES 18 (an end user site, *see also* Application, page 63, line 18, through page 66, line 3) and 20a-20b (an administration site, *see also* Application, page 56, lines 1-8).

The method of Claim 11 includes "extracting at least one object of a component from an individual export file of an exported site, wherein the exported site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects; wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location." Components of a site include file assets and non-file assets encapsulating elements of the site (e.g., logon permissions, administrative permissions, site branding, site content and site navigation, etc.). *See* Application, page 9, line 14, through page 10, line 21. File assets include resources such as code including JSP pages, ASP pages, Java classes and/or object oriented programming language classes, and images including GIF files, etc. *See* Application, page 10, lines 5-7. Non-file assets include instantiated programming language objects, permissions, user preferences and setting such as users, groups, modules, module types, pages, menus, themes, structures, styles and templates. *See* Application, page 9, line 19, through page 10, line 2, and lines 8-11. An individual export file may contain at least one component having at least one non-file asset configured to operate on the system at the remote location. *See* Application, page 3, lines 12-13, and page 4, lines 14-15. Non-file assets may be constructed as an extensible markup language fragment, such as an XML fragment,

having a predetermined structure. *See* Application, page 57, lines 16-19. The XML fragments may be stored as individual files such as component archive ("CAR") files. *See* Application, page 58, lines 5-8, FIGURE 13. Individual CAR files may be collected and stored as a group export file ("TRUCK"). *See* Application, page 58, lines 10-11, FIGURE 13. The export file may be any one of a CAR file or a TRUCK file. *See* Application, page 58, lines 19-20. According to the method of Claim 11, the system at the remote location in the portal framework can import components of a site by importing an export file (e.g., a CAR file) and extracting from the CAR file at least one non-file asset (e.g., an XML fragment) configured to operate on the system at the remote location.

The method of Claim 11 includes "storing each object of the component to a location on the system at the remote location." The assets of the exported component may be extracted from the export file to a plurality of locations on the system at the remote location. *See* Application, page 3, lines 17-18. For example, an XML fragment configured to operate on the system (portal) at the remote location may be extracted to an appropriate location in a file system of the portal (e.g., system 106a). *See* Application, page 59, lines 3-4. The XML fragment is then parsed and the objects contained therein are instantiated in a database (e.g., system 108a) or other relevant location (e.g., system 110a). *See* Application, page 59, lines 4-6.

Claim 16 recites:

> A computer program product embodied in a computer readable medium for deploying a component of a site between systems in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:
>
> designating a component of the site intended for export, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;
>
> collecting at least one object of the component in an individual export file;
>
> wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location;
>
> transferring the individual export file to a system at a remote location; and
>
> extracting each object from the individual export file to a location on the system at the remote location.

Claim 16 recites a computer program product embodied in a computer readable medium for deploying a component of a site between systems in a portal framework. *See* Application, page 4, lines 7-10. As exemplified in FIGURE 1, a portal framework ("Framework") 100 may comprise systems 102, systems 110, systems 106, and systems 108. In a preferred

embodiment, systems 102 implement user systems, systems 110 implement web servers, systems 106 implement application servers, and systems 108 implement database systems. A web server 110 and an application server 106 can be one in the same computer system. *See* Application, page 13, lines 8-9. A site is considered, for the purposes of the present invention, to be a collection of software objects given a single identity. *See* Application, page 9, lines 14-15. The single identity may be characterized by a shared look-and-feel, a shared set of navigation links, and members of a group who are automatically granted privileges to perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 9, lines 15-19. The look-and-feel of a site is referred to, for purposes of the present invention, as the branding for the site. *See* Application, page 51, lines 17-19. The collection of software objects may be managed by a set of users granted privileges associated with respective objects in the collection of software objects. *See* Application, page 8, lines 10-14. Following the example shown in FIGURE 1, Framework 100 may employ and maintain portals to provide gateways for access to the collection of software objects of a site based on granted privileges. *See* Application, page 8, lines 14-15. Through these portals, users with proper privileges can create and manage the collection of software objects of a site and perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 8, lines 10-14, FIGURES 16a-16b.

The computer readable medium of Claim 16 carries computer program instructions, executable by a processor, for performing a step of "designating a component of the site intended for export, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects." A component of the site intended for export may be exported through a deployment type administration. *See* Application, page 56, lines 19-20. Deployment type administration to export a component may include designating a component for export via a graphical user interface. *See* Application, page 56, lines 20-21. An administration site may provide the graphical user interface. *See* Application, page 56, line 21, through page 57, line 1. Administration of sites (e.g., creating a site, deploying a site, locking down elements, etc.) may be performed anywhere in Framework 100. *See* Application, page 34, lines 4-16. For example, a user system 102 may be used to provide for the administration of sites on Framework 100. *See* Application, page 12, lines 4-9. As another example, the presentation of web site and administration of sites objects may be implemented by a system

106. *See* Application, page 13, lines 1-6. The aforementioned branding for the site may apply to all users in Framework 100. *See* Application, page 62, lines 11-13. Examples of site branding are shown in FIGURES 18 (an end user site, *see also* Application, page 63, line 18, through page 66, line 3) and 20a-20b (an administration site, *see also* Application, page 56, lines 1-8). In an embodiment of the present invention, multiple components of a site may be simultaneously designated for export. *See* Application, page 57, lines 1-2.

The computer readable medium of Claim 16 also carries computer program instructions, executable by a processor, for performing a step of "collecting at least one object of the component in an individual export file; wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location." Components of a site include file assets and non-file assets encapsulating elements of the site (e.g., logon permissions, administrative permissions, site branding, site content and site navigation, etc.). *See* Application, page 9, line 14, through page 10, line 21. File assets include resources such as code including JSP pages, ASP pages, Java classes and/or object oriented programming language classes, and images including GIF files, etc. *See* Application, page 10, lines 5-7. Non-file assets include instantiated programming language objects, permissions, user preferences and setting such as users, groups, modules, module types, pages, menus, themes, structures, styles and templates. *See* Application, page 9, line 19, through page 10, line 2, and lines 8-11. These assets may be collected by a subsystem for the site and stored as individual export files. *See* Application, page 57, line 8, through page 58, line 6. An individual export file may contain at least one component having at least one non-file asset configured to operate on the system at the remote location. *See* Application, page 3, lines 12-13, and page 4, lines 14-15. Non-file assets may be constructed as an extensible markup language fragment, such as an XML fragment, having a predetermined structure. *See* Application, page 57, lines 16-19. The XML fragments may be stored as individual files such as component archive ("CAR") files. *See* Application, page 58, lines 5-8, FIGURE 13. Individual CAR files may be collected and stored as a group export file ("TRUCK"). *See* Application, page 58, lines 10-11, FIGURE 13. The export file may be any one of a CAR file or a TRUCK file. *See* Application, page 58, lines 19-20.

The computer readable medium of Claim 16 further carries computer program instructions, executable by a processor, for performing a step of "transferring the individual export file to a system at a remote location." The individual export file (e.g., a CAR file)

containing the at least one non-file asset (e.g., an XML fragment) configured to operate on the system at the remote location may be transmitted over a network to the system at the remote location employing a file transfer protocol, such as FTP. *See* Application, page 58, lines 13-15

The computer readable medium of Claim 16 carries computer program instructions, executable by a processor, for performing a step of "extracting each object from the individual export file to a location on the system at the remote location." The assets of the exported component may be extracted from the export file to a plurality of locations on the system at the remote location. *See* Application, page 3, lines 17-18. For example, an XML fragment configured to operate on the system (portal) at the remote location may be extracted to an appropriate location in a file system of the portal (e.g., system 106a). *See* Application, page 59, lines 3-4. The XML fragment is then parsed and the objects contained therein are instantiated in a database (e.g., system 108a) or other relevant location (e.g., system 110a). *See* Application, page 59, lines 4-6.

Claim 26 recites:

> A computer program product embodied in a computer readable medium for importing components of a site to a system at a remote location in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:
>
> extracting at least one object of a component from the individual export file of an exported site wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location, wherein the exported site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects; and
>
> storing each object of the component to a location on the system at the remote location.

Claim 26 recites a computer program product embodied in a computer readable medium for importing components of a site to a system at a remote location in a portal framework, implementing the method steps of Claim 11 as described above. Thus, independent Claims 11 and 26 as well as their respective dependent Claims 12-15, 27-29, 63-64, and 69-70 stand or fall together.

Claim 30 recites:

> A system for deploying a component of a site between systems in a portal framework, comprising:
>
> a local system configured with a first deployment manager and operable to collect at least one object of a component of a site for deployment as an individual export

file, wherein each object includes at least one non-file asset configured to operate on a system at a remote location;

a means for transferring the individual export file from the local system to the system at the remote location; and

the system at the remote location configured with a second deployment manager and operable to extract each object from the individual export file to a location on the system at the remote location.

Claim 30 recites a system for deploying a component of a site between systems in a portal framework. *See* Application, page 4, line 16, through page 5, line 2. As exemplified in FIGURE 1, a portal framework ("Framework") 100 may comprise systems 102, systems 110, systems 106, and systems 108. In a preferred embodiment, systems 102 implement user systems, systems 110 implement web servers, systems 106 implement application servers, and systems 108 implement database systems. A web server 110 and an application server 106 can be one in the same computer system. *See* Application, page 13, lines 8-9. A site is considered, for the purposes of the present invention, to be a collection of software objects given a single identity. *See* Application, page 9, lines 14-15. The single identity may be characterized by a shared look-and-feel, a shared set of navigation links, and members of a group who are automatically granted privileges to perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 9, lines 15-19. The look-and-feel of a site is referred to, for purposes of the present invention, as the branding for the site. *See* Application, page 51, lines 17-19. The collection of software objects may be managed by a set of users granted privileges associated with respective objects in the collection of software objects. *See* Application, page 8, lines 10-14. Following the example shown in FIGURE 1, Framework 100 may employ and maintain portals to provide gateways for access to the collection of software objects of a site based on granted privileges. *See* Application, page 8, lines 14-15. Through these portals, users with proper privileges can create and manage the collection of software objects of a site and perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 8, lines 10-14, FIGURES 16a-16b.

The system of Claim 30 includes "a local system configured with a first deployment manager and operable to collect at least one object of a component of a site for deployment as an individual export file, wherein each object includes at least one non-file asset configured to operate on a system at a remote location." Administration of sites (e.g., creating a site, deploying a site, locking down elements, etc.) may be performed anywhere in Framework 100.

See Application, page 34, lines 4-16. For example, a user system 102 may be used to provide for the administration of sites on Framework 100. *See* Application, page 12, lines 4-9. As another example, the presentation of web site and administration of sites objects may be implemented by a system 106. *See* Application, page 13, lines 1-6. Thus, Claim 30 recites a system for deploying a component of a site between a local system and a remote system in a portal framework in which the local system exports the component and the remote system imports the component. FIGURE 12 is an exemplary diagram illustrating a deployment management implemented by a deployment management API 1202 within site administration software 314 residing on a memory 308 of a system 106 as shown in FIGURE 3b. In this example, system 106a is the local system from where the component of the site is deployed and system 106b is the remote system which imports the components of the site. Deployment management API 1202a (i.e., the first deployment manager, also referred to as the deployment framework 1202a) of system 106a may employ a set of subsystems 1206a. *See* Application, page 57, lines 8-9. A subsystem may identify components for which it is configured to manage. *See* Application, page 57, lines 9-14. The subsystem may collect assets, such as file assets and non-file assets, of each component they identify. *See* Application, page 57, lines 16-17. Through subsystems 1206a, local system 106a is operable to collect at least one asset (object) of a component of a site for deployment as an individual export file. *See* Application, page 58, lines 5-9. Components of a site include file assets and non-file assets encapsulating elements of the site (e.g., logon permissions, administrative permissions, site branding, site content and site navigation, etc.). *See* Application, page 9, line 14, through page 10, line 21. File assets include resources such as code including JSP pages, ASP pages, Java classes and/or object oriented programming language classes, and images including GIF files, etc. *See* Application, page 10, lines 5-7. Non-file assets include instantiated programming language objects, permissions, user preferences and setting such as users, groups, modules, module types, pages, menus, themes, structures, styles and templates. *See* Application, page 9, line 19, through page 10, line 2, and lines 8-11. The assets of a component identified for export may be collected and stored as individual export files. *See* Application, page 57, line 8, through page 58, line 6. Each asset (object) collected in an individual export file includes at least one non-file asset configured to operate on a system at a remote location. *See* Application, page 3, lines 12-13, and page 4, lines 14-15. Non-file assets may be constructed as an extensible markup language fragment, such as an XML fragment, having a predetermined structure. *See* Application, page 57, lines 16-19. The XML fragments may be stored as individual files such as

component archive ("CAR") files. *See* Application, page 58, lines 5-8, FIGURE 13. Individual CAR files may be collected and stored as a group export file ("TRUCK"). *See* Application, page 58, lines 10-11, FIGURE 13.

The system of Claim 30 also includes "a means for transferring the individual export file from the local system to the system at the remote location." In embodiments of the invention, various transfer means may be utilized for transferring the individual export file (e.g., a CAR file) from the local system (e.g., system 106a) to the system at the remote location (e.g., system 106b). For example, components may be exported between systems over a transportation link 104, as shown in FIGURE 12, or by transfer to a computer readable medium. *See* Application, page 57, lines 1-7.

The system of Claim 30 further includes "the system at the remote location configured with a second deployment manager and operable to extract each object from the individual export file to a location on the system at the remote location." As exemplified in FIGURE 12, the system at the remote location (e.g., system 106b) may implement a deployment management API 1202b similar to the aforementioned deployment management API 1202a. Deployment management API 1202b (i.e., the second deployment manager, also referred to as the deployment framework 1202b) of system 106b may employ a set of subsystems 1206b. *See* Application, FIGURE 12. The individual export file (e.g., a CAR file) may be imported by system 106b through deployment framework 1202b. *See* Application, page 58, lines 16-20, FIGURE 12. Through subsystems 1206b, remote system 106b is operable to extract each object from the individual export file to a location on remote system 106b. *See* Application, page 58, lines 5-9. For example, an XML fragment configured to operate on system 106b at the remote location may be extracted to an appropriate location in a file system of remote system 106b. *See* Application, page 59, lines 3-4. The XML fragment is then parsed and the objects contained therein are instantiated in a database or other relevant location, such as a web server. *See* Application, page 59, lines 4-6.

Claim 43 recites:

> A system for importing a component of a site in a portal framework comprising:
> a local system configured with a component module operable to:
> extract at least one object of the component of the site from an individual export file, wherein each object includes at least one non-file asset configured to operate on the local system, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions

associated with each software object in the collection of software objects, and wherein at least one non-file asset is constructed as an extensible markup language fragment having a predetermined structure;

parse the extensible markup language fragment;
instantiate each non-file asset; and
store each object of the component to the local system.

The system of Claim 43 imports a component of a site in a portal framework and includes "a local system configured with a component module". As exemplified in FIGURE 1, a portal framework ("Framework") 100 may comprise systems 102, systems 110, systems 106, and systems 108. In a preferred embodiment, systems 102 implement user systems, systems 110 implement web servers, systems 106 implement application servers, and systems 108 implement database systems. A web server 110 and an application server 106 can be one in the same computer system. *See* Application, page 13, lines 8-9. A site is considered, for the purposes of the present invention, to be a collection of software objects given a single identity. *See* Application, page 9, lines 14-15. The single identity may be characterized by a shared look-and-feel, a shared set of navigation links, and members of a group who are automatically granted privileges to perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 9, lines 15-19. The look-and-feel of a site is referred to, for purposes of the present invention, as the branding for the site. *See* Application, page 51, lines 17-19. The collection of software objects may be managed by a set of users granted privileges associated with respective objects in the collection of software objects. *See* Application, page 8, lines 10-14. Following the example shown in FIGURE 1, Framework 100 may employ and maintain portals to provide gateways for access to the collection of software objects of a site based on granted privileges. *See* Application, page 8, lines 14-15. Through these portals, users with proper privileges can create and manage the collection of software objects of a site and perform administration on at least some of the software objects in the collection as well as elements of the site. *See* Application, page 8, lines 10-14, FIGURES 16a-16b.

In the example of FIGURE 12, a user at system 106b may wish to import a component of a site. In this case, system 106b is a system local to the user. Deployment management API 1202b of system 106b may employ a set of subsystems 1206b. *See* Application, page 57, lines 8-9. The set of subsystems for the site includes a site subsystem, a module subsystem, a template subsystem and a style subsystem. *See* Application, page 57, lines 10-12. System 106b, through deployment management API 1202b, may query subsystems to identify the

component of the site that the user wishes to import, import the identified component in an export file, and extract at least one object of the component of the site from the export file. *See* Application, page 57, lines 9-14, FIGURE 12. Components of a site include file assets and non-file assets encapsulating elements of the site (e.g., logon permissions, administrative permissions, site branding, site content and site navigation, etc.). *See* Application, page 9, line 14, through page 10, line 21. File assets include resources such as code including JSP pages, ASP pages, Java classes and/or object oriented programming language classes, and images including GIF files, etc. *See* Application, page 10, lines 5-7. Non-file assets include instantiated programming language objects, permissions, user preferences and setting such as users, groups, modules, module types, pages, menus, themes, structures, styles and templates. *See* Application, page 9, line 19, through page 10, line 2, and lines 8-11. Non-file assets may be constructed as an extensible markup language fragment, such as an XML fragment, having a predetermined structure. *See* Application, page 57, lines 16-19. The XML fragments may be stored as individual files such as component archive ("CAR") files. *See* Application, page 58, lines 5-8, FIGURE 13. Individual CAR files may be collected and stored as a group export file ("TRUCK"). *See* Application, page 58, lines 10-11, FIGURE 13. According to the system of Claim 43, each asset (object) extracted from the individual export file (e.g., a CAR file) includes at least one non-file asset (e.g., an XML fragment having a predetermined structure) configured to operate on the local system (e.g., system 106b).

The individual export file (e.g., a CAR file) may be imported by system 106b through deployment framework 1202b. *See* Application, page 58, lines 16-20, FIGURE 12. Through subsystems 1206b, remote system 106b is operable to, in addition to extracting the XML fragment having a predetermined structure, parse the XML fragment, instantiate each non-file asset, and store each object of the component to the local system. *See* Application, page 58, lines 5-9. More specifically, the XML fragment having a predetermined structure configured to operate on system 106b may be extracted to an appropriate location in a file system of remote system 106b and then parsed to instantiate the objects contained therein in a database or other relevant location, such as a web server. *See* Application, page 59, lines 3-6.

Claim 79 recites:

> A computer-implemented method of deploying a component of a functioning application between systems, the method comprising the steps of:
>
> designating a component of the functioning application intended for export, wherein the functioning application comprises a collection of software objects

manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting a set of objects of the component of the functioning application designated for export in an individual export file;

transferring the individual export file to a system at a remote location; and

extracting each object from the individual export file to a location, wherein each object is configured to operate on the system at the remote location.

Claim 79 recites a method of deploying a component of a functioning application between systems in a portal framework, implementing one embodiment of the method steps of Claim 1 as described above in which a site is a functioning application.

Claim 83 recites:

A computer program product embodied in a computer readable medium for deploying a component of a functioning application between systems in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:

designating a component of the functioning application intended for export, wherein the functioning application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting at least one object of the component designated for export as in an individual export file, wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location;

transferring the individual export file to a system at a remote location; and

extracting the assets of the component from the individual export file to a plurality of locations on the system at the remote location.

Claim 83 recites a computer program product embodied in a computer readable medium for deploying a component of a functioning application between systems in a portal framework, implementing the method steps of Claim 79, which implements one embodiment of the method steps of Claim 1 as described above in which a site is a functioning application. Thus, independent Claims 79 and 83 as well as their respective dependent Claims 80-82 and 84-86 stand or fall together.

Claim 87 recites:

A computer-implemented method of deploying a functioning application between systems, comprising:

collecting a set of software components used by a first application on a first system in a set of individual export files, wherein the first application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting a set of configurations for use with the set of software components in the set of individual export files;

collecting the data utilized by the software components in the set of individual export files;

collecting the set of individual export files into a group export file;
transferring the group export file to a second system at a remote location;
extracting the set of individual export files from the group export file;
extracting the set of software components, the set of configurations, and the data from the set of individual export files to create a second application on the second system, wherein the second application has the same state as the first application.

Claim 87 recites a method of deploying a functioning application between systems, implementing one embodiment of the method steps of Claim 1 as described above in which a site is a functioning application. In this case, a set of software components used by a functioning application, which may be executing, on a first system (e.g., system 106a of FIGURE 3b) is collected in a set of individual export files (e.g., CAR files). *See* Application, FIGURES 1, 3b, and 12-13. The first application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects. A set of configurations as well as data (e.g., file assets and non-file assets) for use with the set of software components is also collected in the set of individual export files. According to the method of Claim 87, the set of individual export files is then collected into a group export file (e.g., TRUCK). *See* Application, page 58, lines 10-11. The group export file is transferred to a second system at a remote location (e.g., system 106b). At the remote system, the set of individual export files is extracted from the group export file and the set of software components, the set of configurations, and the data are extracted from the set of individual export files, which encapsulated the data and logical processes of the first application, to create a second application on the second system, wherein the second application has the same state as the first application. *See* Application, page 10, lines 3-21.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

In the Final Office Action dated June 28, 2006, Claims 1, 3-7, 11-22, 26-32, 34-38, 43, 46, and 59-88 were collectively rejected under 35 U.S.C. § 103 as being unpatentable over *Developing Applications with JRun*, Allaire Corp., May 10, 2001 (hereinafter referred to as the "JRun Manual") in view of G. Douglas, *Web Browser File Uploading to EAS Server*, Sybase, pp. 1-5, January 15, 2001 (hereinafter referred to as "Douglas").

The examiner alleged that, as to independent Claim 1, the JRun Manual teaches the invention substantially as claimed by disclosing a method comprising:

Designating a component intended for export, citing p. 381 web application as component;

Storing assets of the component designated in an individual export file, citing p. 381 and 390 and applying either WAR file or JAR file to the individual export file;

Collecting the individual export file into a group export file, citing pp. 408-409 and applying EAR to the group export file;

Extracting the individual export file from the group export file, citing p. 408 in which the "JRun explodes WAR files contained in EAR file and JRIJN deplays EJR JAR files";

Extracting the assets of the component from the individual export file to a plurality of locations on the system at the remote location, citing pp. 409 and 411; and

the JRun Manual teaches a method further comprising collecting the assets of the designated component, citing pp. 381-383, wherein the assets include file assets and non-file assets configured to operate on the system, citing pp. 408-411 "showing expansion of WAR files into directory structure as file assets and deploying the JAR files and application.xmlfile as non-file assets).

The Appellant respectfully submits that the examiner's rejection as set forth above appears to be directed to a previous version of Claim 1 and does not address all the limitations of the present Claim 1, which recites:

> A computer-implemented method of deploying components of a site between systems, the method comprising the steps of:
>
> storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;
>
> transferring the individual export file to a system at a remote location, wherein the system is a web portal capable of executing program instructions; and
>
> extracting the assets from the individual export file to a plurality of locations on the system at the remote location, wherein the assets include file assets and non-file assets configured to operate on the web portal at the remote location.

The examiner further alleged that, as to Claim 1, the JRun Manual teaches that a EAR file/group export file is deployed (p. 408) but does not explicitly teach a method wherein the export file is transferred to a system at a remote location and the extracted assets are stored on the system at the remote location. The examiner cited Douglas as teaching a method wherein the export file is transferred to a system at a remote location and the extracted assets are stored on the system at the remote location, citing pp. 1-5 uploading file from client to server. The examiner alleged that it "would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the JRun Manual's single system to transfer the export file from the client/development system to a remote system/server because it would allow development to occur on a system separate from the remote system/server, thus increasing the reliability of the JRun Manual's web application." The Appellant respectfully submits that there is no support for the alleged motivation to modify the JRun Manual with Douglas.

The examiner appeared to rely on Stanley et al. (U.S. Patent Application Publication No. 20020156756 A1, herein after referred to as "Stanley") for the teaching of using access privileges to prevent unauthorized access, citing paragraph 130 of Stanley. The examiner alleged that "it would have been obvious to one of ordinary skill in the art to modify the previously cited references [JRun Manual and Douglas] with Stanley et al. for the reasons expressly set forth in para. [0130]." Paragraph [0130] of Stanley is reproduced below:

> [0130] Examples of enabling code for the implementation of the object state engine (OSE) 1014 property pane and status management component (SMC) 208 are shown in the following.

The Appellant respectfully submits that there is no support for the alleged motivation to

modify the JRun Manual and Douglas with Stanley.

The above-identified Final Office Action did not contain any substantive rejections against independent Claims 16, 26, 30, 43, 79, 83, and 87 and their dependent Claims 17-22, 27-29, 31-32, 34-38, 46, 59-78, 80-82, 84-86, and 88. Independent Claim 11 was not mentioned or rejected in the above-identified Final Office Action. However, cancelled Claims 23-25, 51-52 were rejected in the above-identified Final Office Action. Moreover, the examiner did not dispute the Appellant's previous arguments presented in the Reply dated January 18, 2006, portions of which remain pertinent to the final rejections as set forth in above-identified Final Office Action.

VII. ARGUMENT

REJECTIONS UNDER 35 U.S.C. §103(a)

1. Introduction

Claims 1, 3-7, 11-22, 26-32, 34-38, 43, 46, and 59-88 were rejected under 35 U.S.C. § 103 as being obvious over the JRun Manual in view of Douglas. The examiner alleged that, as to independent Claim 1, the JRun Manual teaches the invention substantially as claimed in Claim 1 and cited Douglas and Stanley to supplement elements missing from the JRun, namely, a way of uploading file from client to server as disclosed by Douglas and a way of using access privileges to prevent unauthorized access as disclosed by Stanley. The examiner further alleged that it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the cited references.

The Appellant respectfully submits that not only do the JRun Manual in view of Douglas and Stanley fail to teach all the limitations of the invention, but additionally, that the teachings of the JRun Manual in view of Douglas and Stanley references are not combinable and that a person of ordinary skill in the art would not have been motivated, at the time the invention was made, to modify the JRun Manual, Douglas and Stanley as alleged by the Examiner. The issues on appeal are whether the JRun Manual, Douglas and Stanley references teach all the limitations of the Appellant's invention, and whether a person of ordinary skill in the art of the invention (portal frameworks) would have found it obvious to modify the teachings of the JRun Manual, a complete Java application server manual, with Douglas and Stanley to thereby arrive at the invention.

2. Prior Art

The invention of the Application is directed to a method, a system and a computer program product for providing a site as a collection of software web objects that can be manipulated by a set of users having assigned privileges defined by permissions associated with each software web object in the collection of objects. In the "Description of the Prior art" section of the Application, the Appellant describes the manner in which objects of a web site

were previously implemented. Specifically, in the "Description of the Prior art" section of the Application, the Appellant describes a prior art system in which implementing web objects of a web site provided by a portal requires the physical transfer of files corresponding to web objects to systems which desire to implement these web objects. *See* Application, page 2, lines 3-9. Alternatively, an entire web site can be archived in a file and transferred to a system in order to implement that web site on the system. *See* Application, page 2, lines 9-10. Not only are these processes complex, costly and prone to error, but the process requires the manual transfer of code and libraries to a system. *See* Application, page 2, lines 10-19.

The JRun Manual explicitly states that "JRun is a complete Java application server for developing and deploying ... *secure server-side* J2EE applications." (*Welcome to JRun,* JRun Manual, page xix, lines 1-2). In other words, JRun would run on the server side and would not serve on the client side. Thus, the JRun Manual appears to support the development and deployment of J2EE applications in the JRun Java application server environment only. Douglas discloses how to upload a file via a web page and store it on a Servlet server (i.e., EAServer 3.0).

Douglas does not teach how to perform administrations (e.g., deploying a site, locking down elements of the site, creating sites anywhere in a portal framework, granting and revoking privileges to users for performing administration, configuring privileges for sites, logging on to end-user sites, mapping created sites to URL, modifying systems properties, site branding, and sharing objects between sites, etc. *See* Application, page 16, lines 6-11) on the site via the web page, thereby managing components of the site from a remote location. Since neither the JRun Manual nor Douglas describes how to administer or manage components of a site between systems outside the JRun Java application server environment, the combination of the JRun Manual and Douglas would/could not have improved the prior art system of implementing web objects of a web site in a portal framework.

Stanley describes a way of using access privileges to prevent unauthorized access. Given that the primary reference, the JRun Manual, explicitly states that "JRun is a complete Java application server for developing and deploying ... *secure server-side* J2EE applications", one of ordinary skill in the art at the time the invention was made would not rely on Stanley's teaching outside the secure JRun Java application server environment. At best, one of ordinary skill in the art at the time the invention was made may use Stanley's teaching

of access privileges to prevent unauthorized uploading file via a web page, in which case, implementing web objects of a web site provided by a portal would still require the physical selection and transfer of files to systems desired to implement the web objects.

3. Improvement Over Prior Art

The invention of the Application improves upon the prior art by providing a method, a system and a computer program product for deploying components of a site between systems in a portal framework. A site is considered a collection of software objects. Components of a site may be exported to, and imported from, a system at a remote location. The import and export may be executed by performing deployment type administrations. The invention provides several advantages over the prior art. For instance, Administration of sites may be performed anywhere in the portal framework. *See* Application, page 34, lines 4-16. For example, a user system 102 may be used to provide for the administration of sites on Framework 100. *See* Application, page 12, lines 4-9. As another example, the presentation of web site and administration of sites objects may be implemented by a system 106. *See* Application, page 13, lines 1-6. Additionally, the branding for a site may apply to all users or a select group of users in the portal framework. *See* Application, page 62, lines 11-13. Examples of site branding are shown in FIGURES 18 (an end user site, *see also* Application, page 63, line 18, through page 66, line 3) and 20a-20b (an administration site, *see also* Application, page 56, lines 1-8). Further, the collection of software objects of a site can be shared according to a set of privileges, including privileges associated with the repository, a user, or a site, etc. For example, an administrator with privileges with respect to one site may share an object from this site with another site. In the aforementioned prior art system, objects cannot be shared between sites, therefore it is also the case that objects cannot be shared according to a set of privileges.

4. Examiner's / Appellant's Positions Regarding Obviousness

The examiner asserted that the invention would have been obvious to one of ordinary skill in the art, at the time the invention was made, over the combination of the JRun Manual, Douglas, and Stanley. The examiner asserted that the JRun Manual teaches the invention substantially as claimed and that Douglas and Stanley teach the limitations lacking in the JRun Manual. Specifically, the examiner alleged that it "would have been obvious to one of ordinary

skill in the art at the time the invention was made to modify the JRun Manual's single system to transfer the export file from the client/development system to a remote system/server because it would allow development to occur on a system separate from the remote system/server, thus increasing the reliability of the JRun Manual's web application." Moreover, the examiner alleged that "it would have been obvious to one of ordinary skill in the art to modify the previously cited references [JRun Manual and Douglas] with Stanley et al. for the reasons expressly set forth in para. [0130]." The Appellant respectfully submits that the combination of the JRun Manual, Douglas, and Stanley does not teach or suggest all the claim limitations. The Appellant further respectfully submits that there is no support for the alleged motivation to modify the JRun Manual with Douglas and Stanley. Additionally, even if one of ordinary skill in the art at the time the invention was would have been motivated to combine the JRun Manual with Douglas and Stanley, the resulting combination would not have solved the web site implementation problems identified in the present Application.

5. Rejections Under 35 U.S.C. 103(a)

5.1 Examiner's Reasoning

The examiner alleged that the JRun Manual teaches the invention substantially as claimed by citing p. 381 web application as component, citing p. 381 and 390 and applying either WAR file or JAR file to the individual export file, citing pp. 408-409 and applying EAR to the group export file, citing p. 408 in which the "JRun explodes WAR files contained in EAR file and JRIJN deplays EJR JAR files", citing pp. 408-411 and pp. 381-383 and alleging that the JRun Manual teaches assets including file assets and non-file assets configured to operate on the system, "showing expansion of WAR files into directory structure as file assets and deploying the JAR files and application.xmlfile as non-file assets.

The examiner further alleged that the JRun Manual teaches that a EAR file/group export file is deployed (p. 408) but does not explicitly teach a method wherein the export file is transferred to a system at a remote location and the extracted assets are stored on the system at the remote location. The examiner cited Douglas as teaching a method wherein the export file is transferred to a system at a remote location and the extracted assets are stored on the system at the remote location, citing pp. 1-5 uploading file from client to server. The examiner

alleged that it "would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the JRun Manual's single system to transfer the export file from the client/development system to a remote system/server because it would allow development to occur on a system separate from the remote system/server, thus increasing the reliability of the JRun Manual's web application."

The examiner appeared to rely on Stanley for the teaching of using access privileges to prevent unauthorized access, citing paragraph 130 of Stanley. The examiner alleged that "it would have been obvious to one of ordinary skill in the art to modify the previously cited references [JRun Manual and Douglas] with Stanley et al. for the reasons expressly set forth in para. [0130]."

5.2 Flaws In The Examiner's Reasoning

The Appellant respectfully submits that there is no support for the alleged motivation to modify the JRun Manual and Douglas with Stanley. A flaw in the examiner's reasoning is that the JRun Manual describes the development and deployment of J2EE applications in the JRun Java application server environment only. Another flaw in the examiner's reasoning is that Douglas describes uploading a file via a web page for storage on a server and discloses that reassembling the encoded file on the server side is not easy. Yet another flaw in the examiner's reasoning is that Stanley failed to provide a proper motivation to modify the JRun Manual and Douglas. The Appellant will explain below how these flaws failed to make a *prima facie* case of obviousness under 35 U.S.C. § 103(a).

5.3 Examiner Has Failed To Show Suggestion Or Motivation To Combine References

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art. *In re Kotzab,* 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000). See also *In re Lee,* 277 F.3d 1338, 1342-44, 61 USPQ2d 1430, 1433-34 (Fed. Cir. 2002) (discussing the importance of relying on objective

evidence and making specific factual findings with respect to the motivation to combine references); *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). Furthermore, if the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959).

First, the examiner cited p. 381 and 390 of the JRun Manual and applied either WAR file or JAR file to the individual export file of the invention. However, the JRun Manual does not explicitly or implicitly teach transferring either WAR file or JAR file from a client/development system to a remote system/server or vice versa (i.e., from a server/development system to a remote client/user system). The JRun Manual explicitly states that "JRun is a complete Java application server for developing and deploying ... *secure server-side* J2EE applications." (*Welcome to JRun,* JRun Manual, page xix, lines 1-2). In other words, JRun would run on the server side and would not serve on the client side. Consequently, the JRun Java application server would have no need to transfer either WAR file or JAR file from a client to a server or vice versa.

Second, allowing development of J2EE applications using the JRun Java application server to occur on a system separate from the the JRun Java application server seems to go against the very purpose of "developing and deploying ... *secure server-side* J2EE applications." *Id.* As one of ordinary skill in the art would have known at the time the invention was made, server-side Java code executes on a web server rather than on a web client. The JRun Manual specifically lists many benefits of server-side Java (*Benefits of server-side Java*, JRun Manual, Chapter 1, page 4). For example, regardless of the compliance level of clients, server-side application can immediately take advantage of new Java features. In other words, allowing development of J2EE applications to occur on a system separate from the JRun Java application server would compromise the many benefits of server-side Java proclaimed in the JRun Manual. Therefore, it is respectfully submitted that, contrary to the examiner's allegations, one of ordinary skill in the art at the time the invention was made would not have been motivated to modify the JRun Manual's teaching to allow development of J2EE applications to occur on a system separate from the JRun Java application server.

Third, neither the JRun Manual nor Douglas seems to teach or suggest that allowing development of J2EE applications with JRun to occur on a system separate from the JRun Java application server would have or could have increased the reliability of the JRun Manual's web application. The Appellant respectfully submits that one of ordinary skill in the art at the time the invention was made would have had no knowledge on whether the alleged combination of the JRun Manual and Douglas would have or could have increased the reliability of the JRun Manual's web application. The Examiner failed to explain, with reasonable details, on how the alleged combination of the JRun Manual and Douglas would have increased the reliability of the JRun Manual's web application and cites reference(s) that would support the combination of the JRun Manual and Douglas. The above arguments were submitted in the aforementioned Reply dated January 18, 2006, which the examiner did not dispute or respond.

5.4 Not All Limitations Disclosed

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970).

Embodiments of the invention as set forth in the claims offer solutions in deploying components of a web site between systems (e.g., end user systems, web server systems, application server systems, database server systems, etc.), achieving technical capabilities beyond the combination of the JRun Manual, Douglas, and Stanley. The Appellant respectfully submits that the combination of the JRun Manual, Douglas, and Stanley does not disclose all the limitations of the invention. For example, Claim 1 recites:

> A computer-implemented method of deploying components of a site between systems, the method comprising the steps of:
>
> storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;
>
> transferring the individual export file to a system at a remote location, wherein the system is a web portal capable of executing program instructions; and
>
> extracting the assets from the individual export file to a plurality of locations on the system at the remote location, wherein the assets include file assets and non-file assets configured to operate on the web portal at the remote location.

According to Claim 1, the site designated for export comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects. Further, Claim 1 recites, "transferring the individual export file to a system at a remote location." The privileges defined by permissions associated with each software object, which are stored in the individual export file, are transferred along with the assets to the system at the remote location. Douglas's teaching of uploading file from a client to a server via a web page fails to disclose this limitation. As discussed above, the JRun Manual describes the development and deployment of J2EE applications in the JRun Java application server environment only. Thus, even if Douglas discloses how to upload a file via a web page and store it on a Servlet server (i.e., EAServer 3.0), the combined teachings of the JRun Manual and Douglas still fail to disclose "storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects" and "transferring the export file to a system at a remote location, wherein the system is a web portal," as set forth in claim 1.

What is more, as was/is known, Java archive (JAR) and web application archive (WAR) files have very specific definitions and uses. In view of the JRun Manual and Douglas, one of ordinary skill in the art would not be able to deploy components of a site between systems (e.g., between a user system and a web server). More specifically, in view of the JRun Manual, a J2EE application developer might be able to use JRun to develop web applications at the server side. In view of Douglas, an end user might be able to use Sybase to upload a file from the user's computer to a web server for storage or processing. However, without the particulars disclosed in the present Application, one of ordinary skill in the art at the time the invention was made would not know how to develop and deploy components of a site between systems (e.g., portals for the general public, portals for the intranet of a company, portals for the extranet for customers, etc., *see* Fig. 1 and page 1, line 16, to page 2, line 2, of the Specification), "wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects," as set forth in claim 1.

As discussed in the "Description of the Prior Art" section of the Application, the complexity and cost of developing, deploying, administering and continually enhancing portals,

is tremendous. *See* Application, page 2, lines 3-19. At the time the invention was made, web sites were typically developed, deployed, administered, and maintained at the server-side and not available to an intended group of users for commercial use. *See* Application, page 2, lines 13-14. At the time the invention was made, one might be able to write, test, and debug Java applications and applets at a client installed with the required software programs such as the Java Development Kit (JDK) (*see* e.g., *Runtime Environment,* the JRun Manual, Chapter 35, page 405). However, as the JRun Manual explicitly discloses, the *deployment* of the Java applications is at the server-side (e.g., from a JRun server to a web server connected thereto, *see Deploying Web Applications*, the JRun Manual, Chapter 34, page 383 and *Deploying J2EE Applications*, the JRun Manual, Chapter 36, page 411). Therefore, even assuming that a web application developed with the JRun Manual could be a component of a site, one of ordinary skill in the art at the time the invention was made still would not be able to deploy that component "to a system at a remote location," as claimed in claim 1. The above arguments were submitted in the aforementioned Reply dated January 18, 2006, which the examiner did not dispute or respond.

5.5 Examiner Has Failed To Make A Prima Facie Case Of Obviousness Under 35 U.S.C. §103

Because the examiner has failed to show that the prior art references teach or suggest all the claim limitations and that there was a proper suggestion or motivation to combine the JRun Manual, Douglas, and Stanley references, the examiner has failed to meet the criteria set forth in M.P.E.P. 2143 for a *prima facie* case of obviousness. Accordingly, the Appellant requests that the rejection of the claims under 35 U.S.C. §103 be withdrawn.

5.6 Summary

Embodiments of the invention can deploy one or more components of a site (which, in one embodiment, could include components of an entire web site) to a system at a remote location (e.g., a web portal at a user system). The site comprises a collection of software objects that can be manipulated by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects. These users do not need to know how each software object performs its processes and yet they can

administer and maintain objects and elements of a site based on their privileges with respect to the objects and elements of the site in a site context or in a system context. *See e.g.,* Application, page 10, lines 12-20, page 11, lines 7-14, and page 14, line 1, through page 20, line 6. This is one of the many technical advantages that are neither taught nor suggested by the combination of the JRun Manual, Douglas and Stanley references.

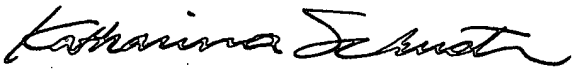
6. Conclusion

As explained above, the Appellant believes that the combination of the JRun Manual, Douglas and Stanley references does not describe each and every limitation of Independent Claims 1, 11, 16, 26, 30, 43, 79, 83, and 87, and that the corresponding rejections of Independent Claims 1, 11, 16, 26, 30, 43, 79, 83, and 87 should properly be withdrawn. Appellant therefore respectfully requests that all of the rejections be withdrawn and that all the pending Claims 1, 3-7, 11-22, 26-32, 34-38, 43, 46, and 59-88 be allowed.

A check in the amount of $500.00 is included with this filing. While Appellant believes no further fees are due and owing, if Appellant is in error, the Commissioner is hereby authorized to deduct the appropriate amount from Deposit Account No. 50-3183 of Sprinkle IP Law Group.

Respectfully submitted,

Sprinkle IP Law Group



Katharina Wang Schuster
Reg. No. 50,000

Date: November 21, 2006

1301 W. 25th Street, Suite 408
Austin, TX 78705
Tel. (512) 637-9220
Fax. (512) 371-9088

VIII. CLAIMS APPENDIX

1. (Previously Presented) A computer-implemented method of deploying components of a site between systems, the method comprising the steps of:

storing assets of at least one component of a site designated for export as an individual export file, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

transferring the individual export file to a system at a remote location, wherein the system is a web portal capable of executing program instructions; and

extracting the assets from the individual export file to a plurality of locations on the system at the remote location, wherein the assets include file assets and non-file assets configured to operate on the web portal at the remote location.

2. (Cancelled).

3. (Previously Presented) The method according to claim 1, further comprising querying a local system to identify one or more components associated with the site.

4. (Previously Presented) The method according to claim 1, wherein each non-file asset is an extensible markup language fragment with a predetermined structure.

5. (Original) The method according to claim 4, further comprising parsing the extensible markup language fragment.

6. (Previously Presented) The method according to claim 5, further comprising instantiating each at least one non-file asset.

7. (Previously Presented) The method according to claim 1, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

8-10. (Canceled).

11. (Previously Presented) A computer-implemented method of importing components of a site to a system at a remote location in a portal framework, the method comprising the steps of:

extracting at least one object of a component from an individual export file of an exported site, wherein the exported site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location; and

storing each object of the component to a location on the system at the remote location.

12. (Previously Presented) The method according to claim 11, further comprising transferring the individual export file to the system at the remote location.

13. (Previously Presented) The method according to claim 11, further comprising parsing each non-file asset wherein each non-file asset is constructed as an extensible markup language fragment with a predetermined structure.

14. (Previously Presented) The method according to claim 11, further comprising instantiating each non-file assets.

15. (Previously Presented) The method according to claim 11, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

16. (Previously Presented) A computer program product embodied in a computer readable medium for deploying a component of a site between systems in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:

designating a component of the site intended for export, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting at least one object of the component in an individual export file;

wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location;

transferring the individual export file to a system at a remote location; and

extracting each object from the individual export file to a location on the system at the remote location.

17. (Previously Presented) The computer program product according to claim 16, further comprising computer program instructions for performing the step of collecting the assets of the component.

18. (Previously Presented) The computer program product according to claim 17, wherein designating the component for export further comprises querying a local system.

19. (Previously Presented) The computer program product according to claim 17, wherein each non-file asset is constructed as an extensible markup language fragment with a predetermined structure.

20. (Previously Presented) The computer program product according to claim 19, further comprising computer program instructions for performing the step of parsing each extensible markup language fragment.

21. (Previously Presented) The computer program product according to claim 20, further comprising computer program instructions for performing the step of instantiating each non-file assets.

22. (Previously Presented) The computer program product according to claim 16, wherein each non-file assets include at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

23-25. (Canceled).

26. (Previously Presented) A computer program product embodied in a computer readable medium for importing components of a site to a system at a remote location in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:

extracting at least one object of a component from the individual export file of an exported site wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location, wherein the exported site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects; and

storing each object of the component to a location on the system at the remote location.

27. (Previously Presented) The computer program product according to claim 26, further comprising transferring the individual export file to the system at the remote location.

28. (Previously Presented) The computer program product according to claim 26, further comprising instantiating each non-file asset.

29. (Previously Presented) The computer program product according to claim 26, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

30. (Previously Presented) A system for deploying a component of a site between systems in a portal framework, comprising:

a local system configured with a first deployment manager and operable to collect at least one object of a component of a site for deployment as an individual export file, wherein each object includes at least one non-file asset configured to operate on a system at a remote location;

a means for transferring the individual export file from the local system to the system at the remote location; and

the system at the remote location configured with a second deployment manager and operable to extract each object from the individual export file to a location on the system at the remote location.

31. (Previously Presented) The system according to claim 30, wherein the object is collected by a first component module.

32. (Original) The system according to claim 31, wherein the first component module is further operable to query the local system to identify the component designated for export.

33. (Canceled).

34. (Previously Presented) The system according to claim 31, wherein the first component module is further operable to construct each non-file asset as an extensible markup language fragment with a predetermined structure.

35. (Previously Presented) The system according to claim 34, wherein the second deployment manager further includes a second component module operable to parse each extensible markup language fragment.

36. (Previously Presented) The system according to claim 34, wherein the second component module is further operable to instantiate each non-file assets.

37. (Previously Presented) The system according to claim 30, wherein the second component module is further operable to extract each object of the component from the export file to a location on the system at the remote location.

38. (Previously Presented) The system according to claim 30, wherein the non-file assets include at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

39-42. (Cancelled).

43. (Previously Presented) A system for importing a component of a site in a portal framework comprising:

a local system configured with a component module operable to:

extract at least one object of the component of the site from an individual export file, wherein each object includes at least one non-file asset configured to operate on the local system, wherein the site comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects, and wherein at least one non-file asset is constructed as an extensible markup language fragment having a predetermined structure;

parse the extensible markup language fragment;

instantiate each non-file asset; and

store each object of the component to the local system.

44-45. (Canceled).

46. (Previously Presented) The system according to claim 43, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

47-58. (Canceled).

59. (Previously Presented) The method according to claim 1, further comprising collecting at least one file-asset of the component in the individual export file, wherein each file-asset is configured to operate on the system at the remote location.

60. (Previously Presented) The method according to claim 59, further comprising querying a subsystem to identify one or more components associated with the site.

61. (Previously Presented) The method according to claim 60, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

62. (Previously Presented) The method according to claim 61, wherein the subsystem collects each object and each file-asset.

63. (Previously Presented) The method according to claim 11, further comprising extracting at least one file-asset of the component from the individual export file; and storing each file-asset to a location on the system at the remote location.

64. (Previously Presented) The method according to claim 63, wherein storing each object and file-asset is accomplished by a subsystem at the remote location, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

65. (Previously Presented) The computer program product according to claim 16, further comprising computer program instructions for collecting at least one file-asset of the component in the individual export file, wherein each file-asset is configured to operate on the system at the remote location.

66. (Previously Presented) The computer program product according to claim 65, wherein designating the component comprises querying a subsystem.

67. (Previously Presented) The computer program product according to claim 66, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

68. (Previously Presented) The computer program product according to claim 67, wherein the subsystem collects each object and each file-asset.

69. (Previously Presented) The computer program product according to claim 26, further comprising

extracting at least one file-asset of the component from the individual export file;

and storing each file-asset to a location on the system at the remote location.

70. (Previously Presented) The computer program product according to claim 69, wherein storing each object and file-asset is accomplished by a subsystem at the remote location, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

71. (Previously Presented) The system according to claim 30, wherein the first deployment manager is further operable to collect at least one file-asset of the component in the individual export file, wherein each file-asset is configured to operate on the system at the remote location.

72. (Previously Presented) The system according to claim 71, wherein the first deployment manager is further operable to query a subsystem.

73. (Previously Presented) The system according to claim 72, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

74. (Previously Presented) The system according to claim 73, wherein the subsystem collects each object and each file-asset.

75. (Previously Presented) The system according to claim 37, wherein the second deployment manager is further operable to:

extract at least one file-asset of the component from the individual export file; and

store each file-asset to a location on the system at the remote location.

76. (Previously Presented) The system according to claim 75, wherein the second deployment manager further includes a subsystem, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

77. (Previously Presented) The system according to claim 43, wherein the component module is further operable to:

extract at least one file-asset of the component from the individual export file; and

store each file-asset to a location on the system at the remote location.

78. (Previously Presented) The system according to claim 77, wherein the component further includes a subsystem, wherein the subsystem is configured to manage a type of component, wherein the type includes at least one of a site, module, template and style.

79. (Previously Presented) A computer-implemented method of deploying a component of a functioning application between systems, the method comprising the steps of:

designating a component of the functioning application intended for export, wherein the functioning application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting a set of objects of the component of the functioning application designated for export in an individual export file;

transferring the individual export file to a system at a remote location; and

extracting each object from the individual export file to a location, wherein each object is configured to operate on the system at the remote location.

80. (Previously Presented) The method of claim 79, wherein the component is a software component used by the application, a configuration of a software component or data used with a software component.

81. (Previously Presented) The method according of claim 80, further comprising instantiating each object that is a non-file asset.

82. (Previously Presented) The method according to claim 80, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a settings object, a menu object, an instantiated programming language object and a user group object.

83. (Previously Presented) A computer program product embodied in a computer readable medium for deploying a component of a functioning application between systems in a portal framework, the computer readable medium carrying computer program instructions, executable by a processor, for performing the steps of:

designating a component of the functioning application intended for export, wherein the functioning application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting at least one object of the component designated for export as in an individual export file, wherein the at least one object includes at least one non-file asset configured to operate on the system at the remote location;

transferring the individual export file to a system at a remote location; and

extracting the assets of the component from the individual export file to a plurality of locations on the system at the remote location.

84. (Previously Presented) The computer program product of claim 83, wherein the component is a software component used by the application, a configuration of a software component or data used with a software component.

85. (Previously Presented) The computer program product according of claim 84, wherein the computer program instructions are further executable for performing the step of: instantiating each object that is a non-file asset.

86. (Previously Presented) The computer program product according to claim 84, wherein each non-file asset includes at least one of: a permissions object, a user preference object, a

settings object, a menu object, an instantiated programming language object and a user group object.

87. (Previously Presented) A computer-implemented method of deploying a functioning application between systems, comprising:

collecting a set of software components used by a first application on a first system in a set of individual export files, wherein the first application comprises a collection of software objects manipulatable by a set of users having assigned privileges defined by permissions associated with each software object in the collection of software objects;

collecting a set of configurations for use with the set of software components in the set of individual export files;

collecting the data utilized by the software components in the set of individual export files;

collecting the set of individual export files into a group export file;

transferring the group export file to a second system at a remote location;

extracting the set of individual export files from the group export file;

extracting the set of software components, the set of configurations, and the data from the set of individual export files to create a second application on the second system, wherein the second application has the same state as the first application.

88. (Previously Presented) The method of claim 87, wherein the first application is executing.

IX. EVIDENCE APPENDIX

Appellant believes that no additional evidence is to be presented.

X. RELATED PROCEEDINGS APPENDIX

Appellant believes that there are no related appeals or interferences.